
Poplar Forms Documentation

Release 6.0.15

Poplar Development

Jul 05, 2022

Contents:

1	Quickstart	3
1.1	Selecting a Notification Template	3
1.2	Selecting a From Content Template	3
1.3	Identifying the Sage Users or Groups	3
1.4	Identifying the Form Actions	4
1.5	Adding the SendApprovalFormEmail Action	4
2	Handler Setup	5
2.1	Configuring a Handler	5
3	Adobe Sign Quickstart	7
3.1	Creating an Account	7
3.2	Creating the API Application	7
4	Leisurely Start	15
5	Poller	17
5.1	Running the Poller with Process Scheduler	17
5.1.1	Setting Up Process Scheduler	17
5.1.2	Setting Up the Windows Scheduled Task	21
5.2	REMOTEACTION.poller	26
6	Form Client	27
7	Workflow Actions	29
7.1	Using Workflow Actions	29
7.2	HTML Content in Forms	29
8	Customize a Form	31
8.1	Create a New Workflow Action	31
8.2	Customize the Form	32
8.3	Adding New Controls	33
8.4	Where's the Data?	33
8.5	Progress To Steps and Form Buttons	33
8.6	Dynamic Initial Values	33
9	Sample Workflows	35
9.1	Adobe Sign - A/P Invoice Batch Approval Workflow	35

The Remote Actions package, and accompanying REMOTEACTION module, are used to integrate Orchid Extender's Workflow for Sage 300 with [fleetingforms.io](#) to securely collect well formatted, validated, data from users or customers from beyond the corporate perimeter.

Note: Remote Action version 6.0 introduces flexible handlers. If you're upgrading or installing new, please have a quick read of the [handlers](#) page.

Want to get up and running with a basic approval form quickly? This guide will walkthrough the steps required to add a remote approval to your workflow.

Adding a remote approval to an existing workflow is easy. This is what needs to be done:

1. Select or create a Message Template to use for the notification to the user that a remote approval is available.
2. Select or create a Message Template to use for the form title and content.
3. Identify the Sage users or Extender User Groups that need to receive a notification.
4. Identify the actions the user can take and how the buttons for those actions should be labelled.
5. Add the `REMOTEACTION.SendApprovalFormEmail` action to your template, providing the email notification Message Template, Users, from content Message Template, and actions as parameters.

That's all! The next time the workflow is triggered, a remote approval form will be created and the users notified.

1.1 Selecting a Notification Template

pass

1.2 Selecting a From Content Template

pass

1.3 Identifying the Sage Users or Groups

pass

1.4 Identifying the Form Actions

pass

1.5 Adding the SendApprovalFormEmail Action

pass

CHAPTER 2

Handler Setup

Newer versions of Remote Action require that at least one handler is setup. Handlers take actions on completed forms. Custom handlers can be developed to take new actions, such as creating an object, in place of the default form approval.

There is one supported handler:

1. Workflow Approval Form handler: advance workflows based on form contents.

Note: If no handlers are setup, you will see errors like this in the log:

```
WARNING  Poller type not configured or failed to start.
```

In cases like this, manually setup the default handler.

2.1 Configuring a Handler

Handlers are configured by adding entries to the Remote Action Handlers custom table. Each entry has a Form Type, which matches the type sent to Fleeting Forms and will be used to dispatch completed forms to the correct handlers. For each type, a dotted class path is provided to the code that will do the handling.

If you're working with workflow approval forms, you need exactly one entry:

- Form Type: `workflow_approval`
- Handler Class: `remote_actions.handlers.workflow_approval.
WorkflowApprovalFormHandler`

Note: The default handler should be added at the time Remote Action is installed. If you're upgrading from a previous version you will have to add it by hand.

If the default handler is not configured it is always safe to add it.

CHAPTER 3

Adobe Sign Quickstart

This guide provides a detailed walkthrough of connecting Remote Action to Adobe Sign.

Note: Adobe Sign is supported in Remote Action version 5.1 and newer.

The overall process that needs to be completed to connect Adobe Sign to the Remote Action service is:

1. Create an account.
2. Create a new API application.
3. Perform the initial configuration in Sage.
4. Complete the OAuth workflow.

3.1 Creating an Account

Remote Actions will connect to the User's Adobe sign account. The user must create an account before a connection can be established.

For demonstration and testing, a free developer account can be used. All agreements created using a demo account have a watermark applied and cannot be used in production.

To create a new developer account, visit the [Adobe Sign Developer Account Sign-Up](#).

3.2 Creating the API Application

An API Application allows a service, like Remote Actions, to obtain a token used to act on a user's behalf. The application must be created and authorized by the user to obtain tokens. The authorization can be revoked at any time.

To create an API application, sign in to Adobe Sign and take the following steps:

1. Once you've signed in, you will be redirected to your dashboard. The URL will be in the form `https://secure.<shard>.adobesign.com/...`. The shard will be a short code like `na1`, `eu2`, or `ap3`. Take note of:

- Shard:



Fig. 1: The shard for this account is `na4`.

2. In the navigation column on the left of the screen, expand *Adobe Sign API* and open *API Applications*.
3. In the API Applications page, click the + icon to create a new application.

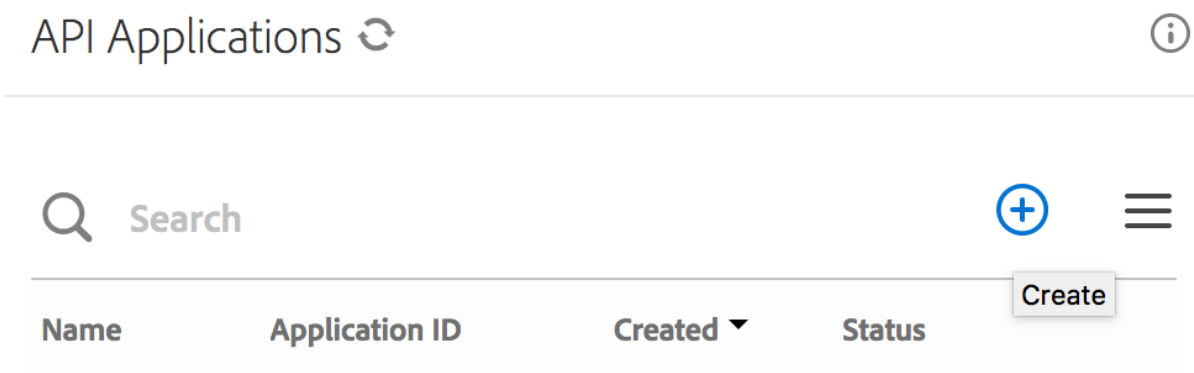


Fig. 2: Click the + icon to create a new application.

4. Create a new CUSTOMER API application.
 - Name: `remote-action-service-<COMPANY>`
 - Description: `Remote Action - Intergate with Sage 300 - <COMPANY>`
 - Domain: `CUSTOMER`
5. Once the application has been created, highlight it and select *Configure OAuth*.
6. Configure OAuth for the application. Take note of:

- Client ID:
- Client Secret:

Set the following values:

- Redirect URI: <https://poplars.dev/adobe-sign-activate.html>
- Enable the following OAuth Scopes with the modifier `self`:
 - `user_read`
 - `agreement_read`
 - `agreement_write`
 - `agreement_send`

5. In Sage, start *Extender* → *Remote Action* → *Setup* → *Adobe Sign Connect*.

Create



Provide a name for your application to issue a set of credentials for use with Adobe Sign's API

Name:

remote-action-service-saminc

Display Name:

Remote Action - Integrate with Sage 300 - SAMINC

Domain:

- ☒ CUSTOMER (This application will only have access to data within your account)
- ☐ PARTNER (This application will have access to any authorized Adobe Sign account)

Cancel

Save

Fig. 3: Define a new CUSTOMER API application.

API Applications

 Search

[View](#) / [Edit](#) | [Deactivate](#) | [Configure OAuth for Application](#)

remote-actions-service-saminc	CBJCHBCAABAA2DFmI5CQFijh9P0Zu7d0vbEVu1sC9Xdp	03/20/2021 05:56	ACTIVE
-------------------------------	--	------------------	--------

Fig. 4: Highlight the new application to reveal *Configure OAuth* link.

Configure OAuth

×

Client ID:

CBJCHBCAABAA2DFmI5CQFiJh9P0Zu7d0vbEVu1sC9Xdp

Client Secret:

_nfpiqyZAv_Ocw4NGz9iqsN-PlfiUCGU

Note: You must keep your Client Secret confidential.

Redirect URI:

https://poplars.dev/adobe-sign-activate.html

Note: The redirectUri specified in your OAuth requests must belong to this list of uris. You can mention multiple uris as comma separated list.

Enabled Scopes

You must enable the scopes that you intend to request through the OAuth protocol. Please limit the scopes that you enable to the minimum set necessary for your application, which is one of the requirements for Certification.

Please [contact support](#) if you need to change which scopes are enabled for your application. ?

Note that only Group Admins can approve OAuth requests that use the ":group" scope modifier, and only Account Admins can approve OAuth requests that use the ":account" scope modifier.

Enabled?	Scope	Modifier	Description
☒	user_read	self	View users in your account
☐	user_write	account	Create or manage users within your account
☐	user_login	account	Login access – providing full access to any user in your account overriding other requests
☒	agreement_read	self	Access documents & data on behalf of any user in your account
☒	agreement_write	self	Manage the status of documents on behalf of any user in your account
☒	agreement_send	self	Send documents on behalf of any user in your account

Fig. 5: Record the Client ID and Client Secret, set the scopes for self.

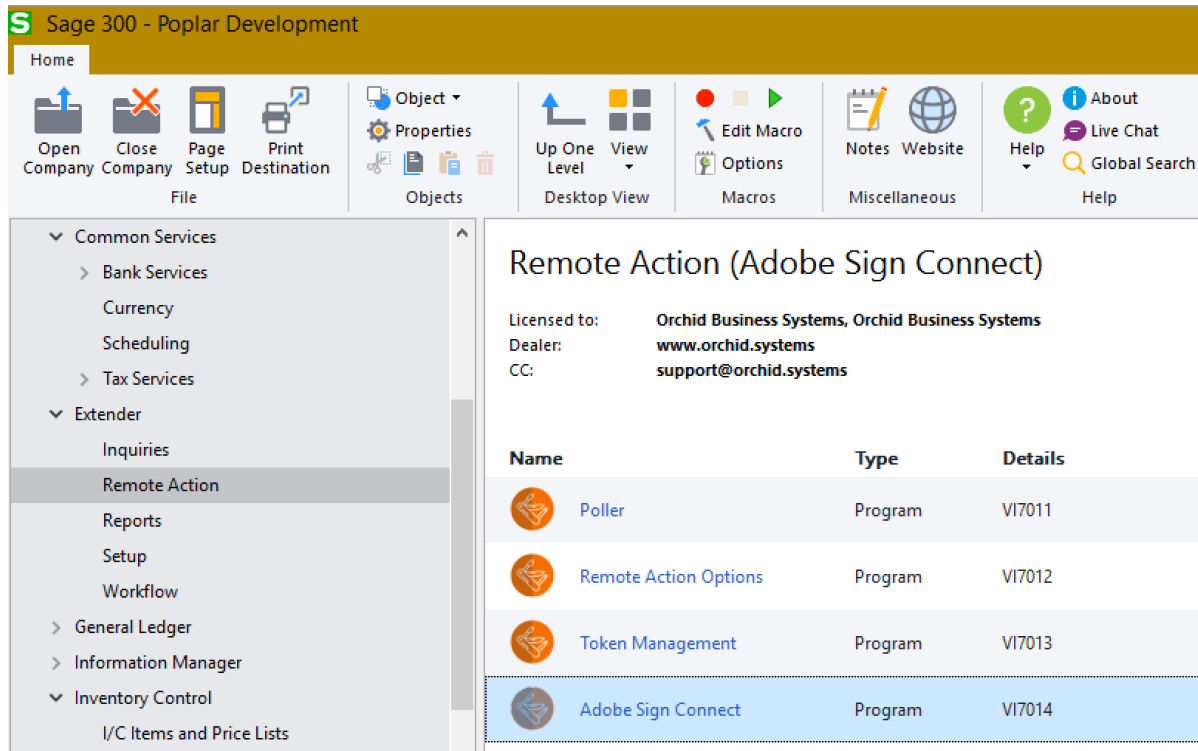


Fig. 6: Open the Adobe Sign Connect utility to start connecting Remote Action to your Adobe Sign account.

6. Fill in the Shard, Client ID, and Client Secret that you noted in the previous steps. Once filled in, the *Authorize* button will be enabled.
7. Click the *Authorize* button. A web browser will open, prompt you to sign in and authorize the application to connect to Adobe Sign on your behalf.
8. After authorizing the application, you will be redirected to the Poplar Development Adobe Sign Connection Confirmation page. This page will display the values required to complete the connection. Take note of:
 - Access Point:
 - Connection Code:
9. Input the Access Point and Connection Code into the Adobe Sign Connect screen in Sage. Complete the connection by clicking on the *Connect* button.

Note: If the access point URL ends with a / it will automatically be removed.

10. Once connected, the configuration is complete.

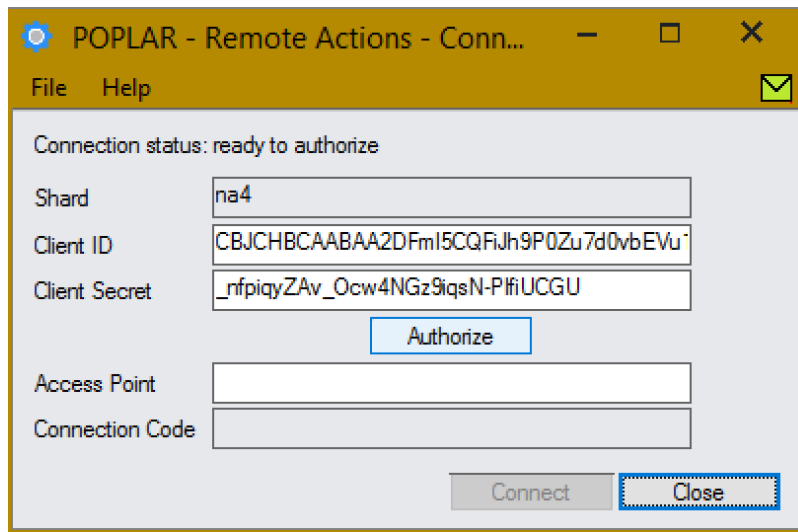


Fig. 7: Input the Shard, Client ID, and Client Secret recorded in the previous steps.

Adobe Sign Activation

Thank you for activating Remote Actions with Adobe Sign.

Your API Access Point is <https://api.na4.adobesign.com/> and the activation code is **CBNCKBAAHBCAABAAA12ka7bMaOe8hbRm4fKDTD03z9e5ukRx**

Copy these values into the Remote Actions Adobe Sign Configuration to complete the connection.

If you have any trouble, please [contact us](#).

Fig. 8: The Poplar Development Adobe Sign Connection Confirmation page displays the Connection Code and API Access point required to complete the connection.

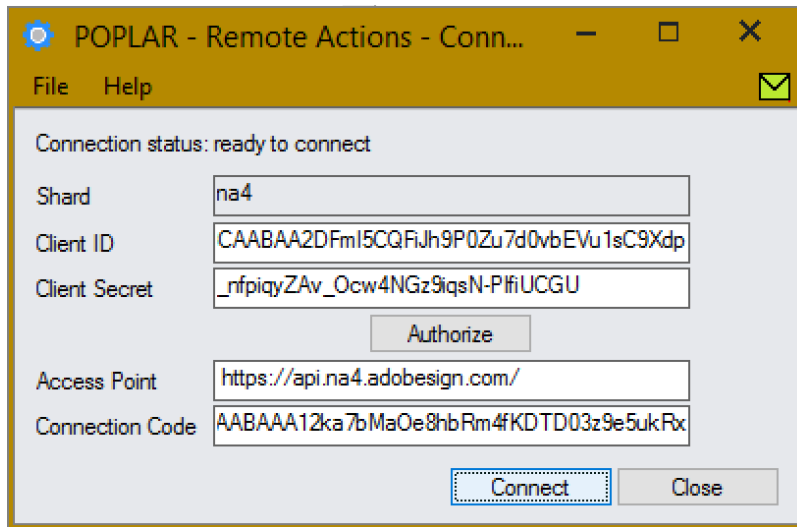


Fig. 9: When you click *Connect*, the process will be completed and a persistent connection established.

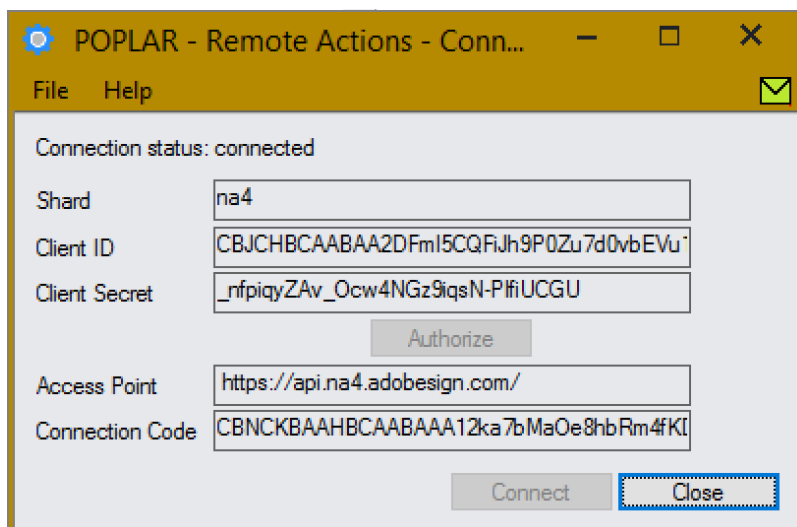


Fig. 10: When you click *Connect*, the process will be completed and a persistent connection established.

CHAPTER 4

Leisurely Start

This guide provides a detailed walkthrough of creating a new workflow that leverages remote approvals.

To provide some context, it is built around a specific use case: approving Sale prices for items.

The poller retrieves completed remote approval forms and applies them to their respective workflows.

The poller can be run manually or using Process Scheduler. Manual execution is helpful while testing and setting up the service but Process Scheduler should be used in production environments.

5.1 Running the Poller with Process Scheduler

The poller is designed to be run in the background as a Task managed by the Windows Task Scheduler. This can be accomplished using Process Scheduler.

5.1.1 Setting Up Process Scheduler

Start by creating an email message template that will be sent to an administrator in the event of an error in the poller.

Open the *Process Scheduler* → *Setup* → *E-Mail Messages* screen. Create a new Process Scheduler Email Template, setting all the required fields and including the details of the errors in the message body.

ORCLTD - Process Scheduler E-mail Messages

File Help

Message ID: PPFORMSPOLL

Description: Error report from the remote approvals poller.

Date Last Maintained: / / ☒ Active

E-mail Subject: Errors in Remote Approvals Poller

The following errors were encountered during the execution of the remote approvals poller:

\$DETAILS

For support with the poller, please contact support@company.com.]

\$SCHEDID - Schedule ID
\$DESC - Schedule Description
\$DETAILS - The integrity check log text
\$AUDIT - The service pack audit log
\$COMPANY_ORGID - Database ID
\$COMPANY_CONAME - Name
\$COMPANY_ADDR01 - Address
\$COMPANY_ADDR02 - Address Line 2
\$COMPANY_ADDR03 - Address Line 3
\$COMPANY_ADDR04 - Address Line 4
\$COMPANY_CITY - City
\$COMPANY_STATE - State/Province
\$COMPANY_POSTAL - Zip/Postal Code
\$COMPANY_COUNTRY - Country
\$COMPANY_LOCTYPE - Location Type
\$COMPANY_LOCCODE - Location Code
\$COMPANY_PHONE - Telephone
\$COMPANY_FAX - Fax Number
\$COMPANY_CONTACT - Contact
\$COMPANY_CNTRYCODE - Country Code
\$COMPANY_BRANCH - Branch
\$COMPANY_HOMECUR - Functional Currency
\$COMPANY_REPORTCUR - Reporting Currency

Add Delete Close

With a template created, it is time for a new Schedule. Open the *Process Scheduler* → *Setup* → *Schedules* screen. Create a new schedule, setting the Schedule ID, description, and email sending parameters.

The screenshot shows the "ORCLTD - Process Scheduler Schedules" application window. The title bar includes standard Windows controls (minimize, maximize, close) and a checkmark icon. The menu bar has "File" and "Help".

The main form displays details for a specific schedule:

- Schedule ID:** PPFORMSPOLL
- Description:** Poll for completed remote approvals.
- Active:** Checked checkbox.
- Send e-mail:** Only if errors (dropdown menu).
- E-mail To:** admin@company.com
- E-mail CC:** (empty field)
- Message ID:** PPFORMSPOLL, with a search icon and an "Open" button.
- Company:** (empty field), with a search icon and a dropdown arrow.

A table below the form lists the steps of the schedule. It has columns: Lin..., Step Number, Company, Action, Send e-mail on error, Next step on error, and Next. The table currently contains no data rows.

At the bottom, there are several action buttons: Add, Delete, Run, Audit Log, Detail..., and Close.

In the Schedules screen, create a new line, Step Number 1, the company for which the poller should run, and set the action to *Run Extender Script*.

ORCLTD - Process Scheduler Schedules

File Help

Schedule ID: PPFORMSPOLL

Description: Poll for completed remote approvals.

☒ Active

Send e-mail: Only if errors

E-mail To: admin@company.com

E-mail CC:

Message ID: PPFORMSPOLL

Company:

Lin...	Step Number	Company	Action	Send e-mail on error	Next step on error	Next step on success
1	1	ORCEWF	Run Extender Script	Yes	0	0

Add Delete Run Audit Log Detail... Close

Open the line details by selecting *Detail...* for the current line. Select the *REMOTEACTION.poller* script and save without any parameters.

Details

×

Line

<< < [1] > >> +

See 'Example Extender Script.py' for an example script.

Step Number

[1]

Action

Run Extender Script ▾

Send e-mail on error

Yes ▾

Company

ORCEWF 🔍

Script

PPFORMS.poller 🔍

Parameters

Next step on error

[0] Send email [🔍]

Next step on success

[0] Send email [🔍]

Date last run

/ /

Time last run

00:00:00

Last status

Number of errors

[0]

Last log

Open

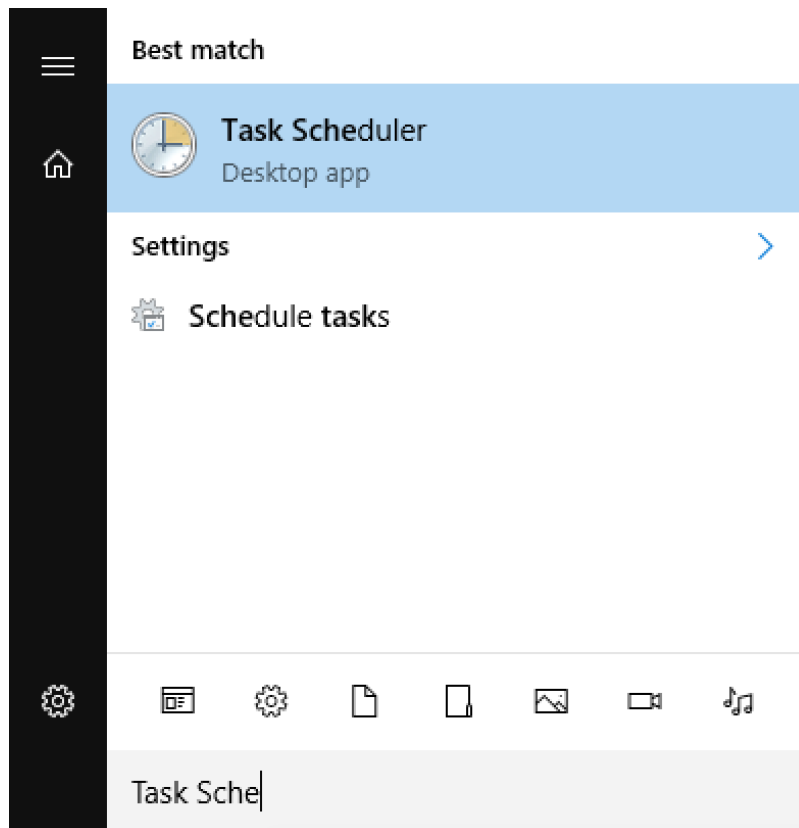
Save

Delete

Close

5.1.2 Setting Up the Windows Scheduled Task

With the schedule configured in Process Scheduler, a scheduled task must be created that can be run automatically in the background. To create a new Scheduled Task, start by opening *Start -> Windows Administrative Tools -> Task Scheduler*.



Right click on the *Task Scheduler Library* and select *Create Task*.

Start by defining the task name, User to run the task, and allow the task to run in the background when the user is not log in.

Poll Remote Actions - Sage 300 Properties (Local Computer)

General | Triggers | Actions | Conditions | Settings | History

Name: Poll Remote Actions - Sage 300

Location: \

Author: ORCHIDV2020\Administrator

Description:

Security options

When running the task, use the following user account:
 ORCHIDV2020\Administrator Change User or Group...

☐ Run only when user is logged on

☒ Run whether user is logged on or not

☐ Do not store password. The task will only have access to local computer resources.

☐ Run with highest privileges

☐ Hidden Configure for: Windows Vista™, Windows Server™ 2008

OK Cancel

Move on to configure the *Triggers*. Create a trigger that will fire when the machine is started and every 5 minutes thereafter indefinitely.

Edit Trigger [X]

Begin the task: At startup ▾

Settings

No additional settings required.

Advanced settings

☐ Delay task for: 15 minutes ▾

☒ Repeat task every: 5 minutes ▾ for a duration of: Indefinitely ▾

☐ Stop all running tasks at end of repetition duration

☐ Stop task if it runs longer than: 3 days ▾

☐ Activate: 5/31/2020 [calendar] 3:34:10 PM [time] ☐ Synchronize across time zones

☐ Expire: 5/31/2021 [calendar] 3:34:10 PM [time] ☐ Synchronize across time zones

☒ Enabled

OK Cancel

Configure the *Action* that will be performed. Create a new action that *Start a program*. The program to start is `OzIntegrityCheck.exe`, located in the `Sage300\oz6?a\` folder. The Schedule ID (`REMOTEACTIONPOLL` in the example) must be provided as an argument.

Edit Action

You must specify what action this task will perform.

Action: Start a program

Settings

Program/script:

C:\Sage\Sage300\oz67a\OZIntegrityCheck.exe Browse...

Add arguments (optional): PPFORMSPOLL

Start in (optional):

OK Cancel

Finally, configure a sensible set of options for the task. The task should rarely take longer than a minute to run. In addition to allowing it being run on demand for testing, it should be stopped automatically if it has run for an hour and multiple instances should not run in parallel.

The screenshot shows a Windows-style dialog box titled "Poll Remote Actions - Sage 300 Properties (Local Computer)". It has a close button (X) in the top right corner. Below the title bar is a tabbed interface with five tabs: "General", "Triggers", "Actions", "Conditions", and "Settings" (which is currently selected). Below the tabs is a large text area with the instruction "Specify additional settings that affect the behavior of the task." followed by several configuration options:

- ☒ Allow task to be run on demand
- ☒ Run task as soon as possible after a scheduled start is missed
- ☐ If the task fails, restart every: 1 minute ▾
- Attempt to restart up to: 3 times
- ☒ Stop the task if it runs longer than: 1 hour ▾
- ☒ If the running task does not end when requested, force it to stop
- ☐ If the task is not scheduled to run again, delete it after: 30 days ▾

Below these options is the text "If the task is already running, then the following rule applies:" followed by a dropdown menu currently showing "Queue a new instance ▾". At the bottom right of the dialog are "OK" and "Cancel" buttons.

Once the configuration of the task is complete, restart the computer for the schedule to take effect, or start it manually.

5.2 REMOTEACTION.poller

CHAPTER 6

Form Client

The form client is a support library that includes the `FormClient` class as well as helper functions that are used across multiple Workflow Actions.

This library is of interest only to those that need to compose custom workflow actions that create, read, or delete approval forms.

Workflow Actions

Included in the Poplar Forms package are a number of workflow actions that can be used to generate and send forms to Sage Users, Extender groups, and client email addresses.

7.1 Using Workflow Actions

The workflow actions included with the package can be used in any workflow step. All workflow actions perform the following:

1. Render the form content message template.
2. Use the rendered content to generate the approval form.
3. Obtain the approval form URL and, if necessary, append default values as URL parameters.
4. Render and send an email notification to all specified users.

In order to perform these tasks, the actions all accept the following parameters:

Parameter1: Notification Email Template - used to render notification email template, inserting the {FORMURL} template variable.
Parameter2: To list - Sage Users or Extender Groups to notify of the approval.
Parameter3: Form Content Template - used to render the form content, the subject is used as the page title and the content is displayed above the form. Both plain text and HTML content are supported.
Parameter4: Button Labels and Progress To steps - a comma separated list of button_label=progress_to_stepname pairs.

7.2 HTML Content in Forms

HTML can be used in form titles and content. Only a subset of HTML tags are supported `ALLOWED_HTML_TAGS`.

To use HTML content in a form, create a message template that represents a valid HTML page (i.e. begins/ends with `<html>` tags and contains a `<body>`).

The following message template will render the customer name in italics, an amount in bold, contains a list, and whitespace management:

```
<html>
  <head></head>
  <body>
    <p>Customer <i>{IDCUST} {NAMECUST}</i>
      has requested a change of credit limit
      from {AMTCRLIMIT:2} to
      <b>{TOVALUE:2}</b>
    </p>

    <br />

    This is a list of things!
    <ul>
    <li>Thing1</li>
    <li>Thing2</li>
    <li>Lorax</li>
    </ul>

    <p>The change was done on {DD}/{MM}/{YYYY} at
      {HOUR}:{MINUTE}:{SECOND} by {USER}</p>
  </body>
</html>
```

Customize a Form

Forms are highly customizable! Many different fields types are supported, validations can be applied, and everything in the form template down to the field help text and error messages are easily changed.

Each Workflow Action contains a list of form controls and an action map at the top of the file after the import statements. By customizing the `form_controls` the layout, fields, and actions of a form can be customized.

For a detailed walkthrough of creating a new action for a custom Sale Price approval workflow based on the `SendApprovalFormEmail` action check out the [leisurely start guide](#).

8.1 Create a New Workflow Action

The first step to customizing a form is copying one of the existing actions to a new Workflow action file. Because workflow actions are stored in the database, an existing action file cannot be copied to a new file and edited.

One method of creating and registering new workflow actions is to view the file contents from the database and copy them into a new file.

1. Open the *Extender* → *Setup* → *Scripts* screen.
2. Right-click on the script that will be used to build the new action and select *View*. The action file contents open in a text editor.
3. In the text editor, select *File* → *New*.
4. Copy the contents of the existing action file to the new file.
5. Close the existing file.
6. Save the new file with a unique name.

8.2 Customize the Form

Each workflow action file included with Poplar Forms has a list of form controls defined at the top of the action file. A form control is a widget that displays a field in the form. The standard approval form template looks something like this:

```
form_controls = [
    {
        'name': 'APPROVALCOMMENT',
        'type': 'textarea',
        'label': 'Comments',
        'required': True,
    }
]
```

To customize the form, change the contents of the `form_controls` variable to meet your needs.

Need a form with a comment that uses a text field instead of a text area? Edit the `APPROVALCOMMENT` form control to be of type `text`:

```
form_controls = [
    {
        'name': 'APPROVALCOMMENT',
        'type': 'text',
        'label': 'Comments',
        'required': True,
    }
]
```

Users sometimes need a nudge, if you want to tell users up front that the comment is required, add some help text to render with the field:

```
form_controls = [
    {
        'name': 'APPROVALCOMMENT',
        'type': 'text',
        'label': 'Comments',
        'required': True,
        'help_text': 'Comments are required, please leave one.',
    }
]
```

Because Comments are required, perhaps an initial value is in order to save time in the best case:

```
form_controls = [
    {
        'name': 'APPROVALCOMMENT',
        'type': 'text',
        'label': 'Comments',
        'required': True,
        'help_text': 'Comments are required, please leave one.',
        'initial': 'Approved'
    }
]
```

8.3 Adding New Controls

Simply define a new element in the `form_controls` list to add an additional control to the form.

What if we need a new form to store the maximum number of widgets that are being approved?

```
form_controls = [
    {
        'name': 'APPROVALCOMMENT',
        'type': 'text',
        'label': 'Comments',
        'required': True,
        'help_text': 'Comments are required, please leave one.',
        'initial': 'Approved'
    },
    {
        'name': 'MAXWIDGETS',
        'type': 'integer',
        'label': 'Maximum Widgets',
        'required': True,
        'help_text': 'What is the maximum number of widgets being approved?',
        'initial': '5432'
    }
]
```

8.4 Where's the Data?

Yes but... where does the data from the new field go? When the Poller applies a form of type `workflow_approval` (like this one) all the fields are set as values in the Workflow itself.

So in the case described above, in the steps following the approval wait step, the `MAXWIDGETS` value will be set in the Workflow and can be referenced as `{MAXWIDGETS}`.

8.5 Progress To Steps and Form Buttons

The buttons displayed at the bottom of the form, and the step that the workflow will progress to when they're pressed, are passed as the fourth parameter to the workflow action.

They are passed as a comma separated list of button label, progress to step name pairs. To render three buttons, *Allow*, *Cancel*, *Deny* that progress to steps *Allowed*, *Canceled*, and *Denied*, provide the following argument:

```
Allow=Allowed,Cancel=Canceled,Deny=Denied
```

8.6 Dynamic Initial Values

There are three ways to set initial values in the form. The first is to include the `initial` argument to the form control definition at the top of the action file:

```
form_controls = [
    {
```

(continues on next page)

(continued from previous page)

```

        'name': 'mycontrol',
        ...
        'initial': 'my initial value'
    }
]

```

Using this approach, all users of the form will see the same default value, regardless of the state of the workflow or the user accessing the form.

To display a value that changes based on the state of the workflow or views but is the same for all users, change the call to `create_workflow_approval_form()` to include the initial value as a keyword argument.

Continuing the MAXWIDGETS example, the initial value can be set to the MAXWIDGETS value in the workflow values by changing the create form call in our custom action:

```

# Create the form
try:
    title, content = render_title_and_content_for(e.resolve(e.p3), e)
    form = create_workflow_approval_form(
        e.wi.viworkih.get("WIID"),
        form_controls,
        title,
        content,
        actions,
        MAXWIDGETS=e.wi.getValue("MAXWIDGETS"))
except Exception as err:
    showMessageBox("Failed to create approval form: {}".format(err))
    return 1

```

The final approach is to set a different initial value per user. This is done by customizing the URL parameters in the link sent to each user. If we wanted to provide a different initial value to each user, we can do so before the email is sent:

```

for (username, email_address) in users:
    email = Email()
    email.setTo(email_address)
    if email.load(e.resolve(e.p1)) == False:
        error("Unable to load message template '" + e.p1 + "'")
        return 1

    user_url = "{}?MAXWIDGETS={}&".format(url, user_max_widgets)
    email.replace("FORMURL", user_url)

```

Note: If more than one approach is used to set the initial value of a form control, only one will be applied.

Initial values set in the form control definition are preferred over those set dynamically.

Initial values set when creating the workflow form are preferred over those set in the URL parameters.

Sample Workflows

This page contains sample workflows for remote action that you can download and try for yourself.

Note: All the workflows on this page will need to be adjusted for your environment before they will work correctly.

[Contact us](#) if you'd like a demo.

9.1 Adobe Sign - A/P Invoice Batch Approval Workflow

This workflow uses Adobe Sign to obtain signatures for the approval of A/P Invoice Batches.

A/P Invoice Batch Approval Template

The package contains [Workflow actions](#) for Orchid Extender that make it simple to create an approval form that can be completed from any computer, phone, or tablet connected to the internet.

This documentation includes information on using the included Workflow actions to [create remote approval forms](#), configuring [handlers](#), setting up the [poller](#), a description of the [form client](#) that is used to connect to [fleetingforms.io](#), and instructions on how to [customize a form](#) to meet your needs.

If you want to get up and running as quickly as possible, check out the [quickstart](#) guide.

Looking for a comprehensive walkthrough of creating a workflow built around a remote approval? Try the [leisurelystart](#).

CHAPTER 10

Indices and tables

- `genindex`
- `modindex`
- `search`